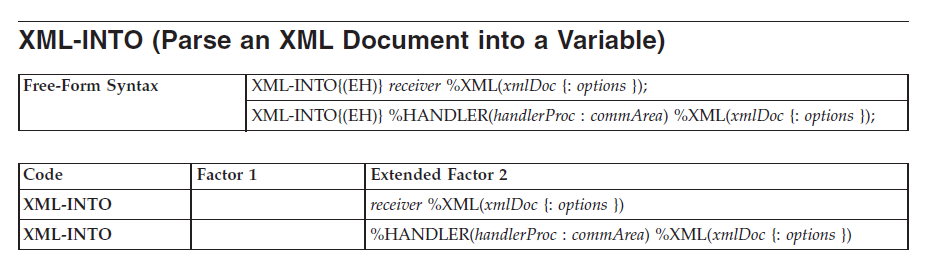
XML inlezen met RPG

Voor het verwerken van XML bestaan er in RPG twee commando’s: XML-INTO en XML-SAX. Het verschil is dat XML-INTO geen gebruik kan maken van doctypes en XML-SAX wel. XML-INTO kan XML-code inlezen in een variabele of in een handler die de XML stapsgewijs inleest. XML-SAX kan alleen via een handler lopen. We concentreren ons hier op XML-INTO.

De syntax is als volgt:



De variabele die met XML gevuld wordt (receiver) kan een los veld zijn of een data structure. Standaard wordt de naam van de variabele gebruikt om het betreffende element in te lezen. Hier kan van afgeweken worden door het pad binnen de XML op te geven. Het pad is het beginelement waar je in de XML wil binnenkomen, bijvoorbeeld Bestelling/BestelHeader. De naam van de data structure is vrij, maar de namen van de subvariabelen moeten exact overeenkomen met de namen van de elementen en hun attributen. De uitzondering geldt hier voor variabelen zonder subvariabelen oftewel ‘children’ in de XML. In dat geval maak je van de variabele alsnog een data structure en verschuift de verplichting van exact dezelfde naam van subvariabele naar data structure met attributen en een waarde waarvan de naam vrij gekozen mag worden.

Laten we ervan uitgaan dat we met data structures werken. Het idee is om de benodigde XML in een data structure in te lezen met de benodigde subvelden. Om de XML te lezen hebben we de inhoud van het hele XML-bestand nodig om door de parser heen te komen. De eerste parameter van de %XML-functie (xmlDoc) kan een string zijn of de naam van een XML-document. Om meerdere levels van elementen in te kunnen lezen, moet je meerdere data structures aanmaken, om deze te kunnen nesten (met het likeds keyword). Let hierbij op dat de data structures van de lagere levels slechts definities zijn om dit mogelijk te maken en niet gevuld worden. Uiteindelijk zal alleen de data structure gevuld worden die als receiver is opgegeven bij het XML-INTO commando. Dit is ook de reden dat eventuele dimensies op het niveau van de in te lezen data structure opgegeven moeten worden en niet op het niveau van de definitie.

Voorbeeld:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| d Interface | ds |  |  |  | qualified |
| d DocType |  | 999a |  |  | varying |
| d DocSoort |  | 999a |  |  | varying |
| d PrijsWijzigingen |  |  |  |  | likeds(PrijsWijzigingen) inz(\*likeds) |
| \* | | |  |  |  |
| d PrijsWijzigingen | ds |  |  |  | qualified |
| d PrijzenHeader |  |  |  |  | likeds(PrijzenHeader) inz(\*likeds) |
| d PrijsDetail |  |  |  |  | likeds(PrijsDetail) inz(\*likeds) dim(999) |
| \* | | |  |  |  |
| d PrijzenHeader | ds |  |  |  | qualified |
| d InkoopNummer |  | 10s | 0 |  | inz(0) |
| \* | | |  |  |  |
| d PrijsDetail | ds |  |  |  | qualified |
| d RegelNummer |  | 3s | 0 |  | inz(0) |
| d ArtikelNummer |  | 999a |  |  | varying |
| d InkoopPrijs |  | 7s | 4 |  | inz(0) |
| d Prijstype |  | 999a |  |  | varying |

Omdat alfanumerieke velden die te lang zijn geen parser-fouten geven, worden deze met een te grote lengte opgegeven, waarna we deze zelf moeten controleren op lengte. Numerieke velden worden wel door de parser gecontroleerd. Als een element meerdere keren kan voorkomen, moeten we een dimensie voor dat element opgeven. Als de variabele die we vullen een array is, wordt er in de PSDS op posities 372-379 het aantal ingelezen elementen gezet.

Om de parser enigszins te sturen kun je gebruik maken van een aantal opties:

* De eerder genoemde optie ‘pad’ wordt gebruikt om aan te geven welk element van de XML gebruikt moet worden. De elementen van het pad worden gescheiden door een slash, bijvoorbeeld path=Bestelling/BestelDetail.
* Om aan te geven dat de parameter xmlDoc van de %XML-functie een bestandsnaam is, moet de optie doc=file worden opgegeven.
* Als het ccsid van het invoerbestand anders is dan op de IBM i, moet je dit opgeven, bijvoorbeeld ccsid=1232.
* De optie case geeft aan of er onderscheid moet zijn tussen kleine en grote letters, bijvoorbeeld case=any (= geen onderscheid).
* De optie trim geeft aan of blanks, tabs en line-ends verwijderd moeten worden voordat de data door de parser aan de variabelen wordt aangeboden.
* De optie allowmissing geeft aan of bepaalde elementen die gedefinieerd zijn in de variabelen, in de XML mogen ontbreken, bijvoorbeeld allowmissing=no.
* De optie allowextra geeft aan of bepaalde elementen die niet gedefinieerd zijn in de variabelen, in de XML mogen voorkomen, bijvoorbeeld allowextra=yes.
* De opties allowextra en allowmissing gelden alleen voor subvelden van de data structure en niet voor de data structure zelf. Dit is zeer belangrijk om te beseffen als je met arrays werkt. In dat geval hoeft niet de hele array gevuld te zijn.
* Wanneer een element attributen heeft, moeten deze ook in de data structure van het betreffende element opgenomen worden. Het laatste subveld van de data structure betreft dan de werkelijke waarde van het element. Dit moet dan als extra optie aangegeven worden als datasubf=veldnaam.
* Met de optie countprefix=prefix kan een prefix worden opgegeven voor het veld binnen de data structure dat het aantal keren telt dat het veld werkelijk voorkomt binnen de dimensie die is opgegeven voor de data structure. De veldnaam wordt dan bijvoorbeeld count\_regelnummer.
* Als er in het XML-bestand namespaces voorkomen, kan de optie ns=optie worden gebruikt. Waarden zijn merge of remove. Merge betekent dat de namespace gebruikt wordt in de veldnamen, de colon(:) van de namespace moet vervangen worden door een underscore(\_). Remove betekent dat de namespace verwijderd wordt in de veldnamen.

De volgende fouten kunnen ontstaan tijdens het parsen:

**00351** Error in XML parsing Algemene parser fout

**00352** Invalid XML option Eén van de opties is niet geldig

**00353** XML document does not match RPG variable Er is iets mis met de gebruikte variabelen

**00354** Error preparing for XML parsing Fout bij de voorbereiding, bijvoorbeeld als een bestand niet bestaat.

Voor deze fouten worden de volgende waarden in de PSDS gevuld:

Posities 43-46 parser error nummer

Posities 91-170 parser error tekst

Posities 368-371 parser return code

Zie voor parser errors en reason codes in bijlage.

De tekens in de volgende tabel mogen niet in de waarden van de elementen voorkomen. Zij moeten vervangen worden door de XML escape code. Bij inlezen worden deze escape codes automatisch door de parser omgezet in het bijbehorende teken.

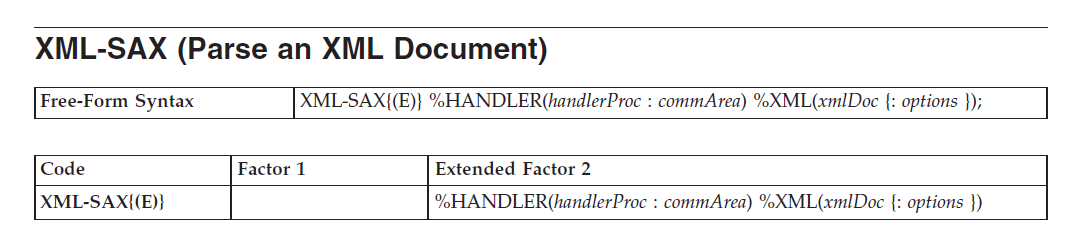
|  |  |
| --- | --- |
| **Teken** | **XML** |
|  | **escape code** |
| & | &amp; |
| " | &quot; |
| ' | &apos; |
| < | &lt; |
| > | &gt; |

Opmerkingen:

* In een XML-bestand mag maar één root voorkomen, het hoogste level. Helaas geeft de parser van de IBM i hier geen fout op als er meerdere zijn. De parser kent eenvoudigweg de opvolgende paden niet.
* De parser kent geen controle tegenover een zogenoemd definitiebestand, zoals een DTD of XSD. Hiermee zijn controles mogelijk als data type, lengte en inhoud. Deze zullen we dus zelf moeten verzorgen.

XML-SAX

Als de noodzaak bestaat om de verwerking wat flexibeler te laten verlopen, moet je het XML-SAX commando gebruiken. Dit kan bijvoorbeeld noodzakelijk zijn als je van tevoren niet weet wat de inhoud van de XML zal zijn. De afhandeling van de parsing ligt nu geheel bij de programmeur. Deze methode is ook handig als je moet ingrijpen in het parsing proces.



Het XML-SAX commando maakt gebruik van een zogenoemde handler. Dit is een van tevoren gedefinieerd prototype met de bijbehorende parameters. Elke keer dat er een event (gebeurtenis) optreedt, zal deze opgeroepen worden. Eén van deze parameters is de communication area die opgegeven wordt in het commando:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| d saxHandler | pr | 10i | 0 |  |  |
| d commArea |  |  |  |  | like(commArea) |
| d event |  | 10i | 0 |  | value |
| d stringPtr |  | \* |  |  | value |
| d stringLen |  | 20i | 0 |  | value |
| d exceptionId |  | 10i | 0 |  | value |
|  | |  |  |  |  |
| d commArea | s | 1a |  |  |  |

De communication area wordt gebruikt om te communiceren tussen de aanroepende procedure en de handler of tussen de verschillende events in.

De event geeft aan wat voor soort gebeurtenis er optreedt, dat kan bijvoorbeeld een nieuwe tag zijn of een parser-fout. Zie voor de verschillende event codes in de bijlage.

De string pointer geeft het adres aan van de string die binnenkomt. De bijbehorende string kan opgevraagd worden via het keyword ‘based’, zie de interface hieronder.

De string lengte geeft aan hoe lang de string is, welke gebruikt kan worden als deze uit het geheugen gelezen wordt.

De exception Id geeft aan wat de eventuele parser-fout is die optreedt.

De interface ziet er vervolgens zo uit:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| p saxHandler | b |  |  |  |  |
| d | pi | 10i | 0 |  |  |
| d commArea |  | 1a |  |  |  |
| d event |  | 10i | 0 |  | value |
| d stringPtr |  | \* |  |  | value |
| d stringLen |  | 20i | 0 |  | value |
| d exceptionId |  | 10i | 0 |  | value |
|  |  |  |  |  |  |
| d string | s | 500a |  |  | based(stringPtr) |
|  |  |  |  |  |  |
| …code (c-specs) |  |  |  |  |  |
|  |  |  |  |  |  |
| c | return 0 |  |  |  |  |
|  |  |  |  |  |  |
| P | e |  |  |  |  |

Voorbeeld voor het uitlezen van een string uit het geheugen:

Eval interface = %subst(string:1:stringLen)

Dit is op deze manier nodig, omdat je van tevoren niet weet hoe lang de variabele zal worden en het geheugen anders verder uitgelezen zal worden.

**Bijlage**

**0351 Parser Error Codes**

|  |  |
| --- | --- |
| **Error Code** | **Description** |
| 1 | The parser found an invalid character while scanning white space outside element content. |
| 2 | The parser found an invalid start of a processing instruction, element, comment, or document type declaration outside element content. |
| 3 | The parser found a duplicate attribute name. |
| 4 | The parser found the markup character '<' in an attribute value. |
| 5 | The start and end tag names of an element did not match. |
| 6 | The parser found an invalid character in element content. |
| 7 | The parser found an invalid start of an element, comment, processing instruction, or CDATA section in element content. |
| 8 | The parser found in element content the CDATA closing character sequence ']]>' without the matching opening character sequence ''. |
| 9 | The parser found an invalid character in a comment. |
| 10 | The parser found in a comment the character sequence '--' (two hyphens) not followed by '>'. |
| 11 | The parser found an invalid character in a processing instruction data segment. |
| 12 | A processing instruction target name was 'xml' in lowercase, uppercase or mixed case. |
| 13 | The parser found an invalid digit in a hexadecimal character reference. |
| 14 | The parser found an invalid digit in a decimal character reference (of the form &#dddd;). |
| 15 | A character reference did not refer to a legal XML character. |
| 16 | The parser found an invalid character in an entity reference name. |
| 17 | The parser found an invalid character in an attribute value. |
| 18 | The parser found a possible invalid start of a document type declaration. |
| 19 | The parser found a second document type declaration. |
| 20 | An element name was not specified correctly. The first character was not a letter, '\_', or ':', or the parser found an invalid character either in or following the element name. |
| 21 | An attribute was not specified correctly. The first character of the attribute name was not a letter, '\_', or ':', or a character other than '=' was found following the attribute name, or one of the delimiters of the value was not correct, or an invalid character was found in or following the name. |
| 22 | An empty element tag was not terminated by a '>' following the '/'. |
| 23 | The element end tag was not specified correctly. The first character was not a letter, '\_', or ':', or the tag was not terminated by '>'. |
| 24 | The parser found an invalid start of a comment or CDATA section in element content. |
| 25 | A processing instruction target name was not specified correctly. The first character of the processing instruction target name was not a letter, '\_', or ':', or the parser found an invalid character in or following the processing instruction target name. |
| 26 | A processing instruction was not terminated by the closing character sequence '?>'. |
| 27 | The parser found an invalid character following '&' in a character reference or entity reference. |
| 28 | The version information was not present in the XML declaration. |
| 29 | The 'version' in the XML declaration was not specified correctly. 'version' was not followed by '=', or the value was missing or improperly delimited, or the value specified a bad character, or the start and end delimiters did not match, or the parser found an invalid character following the version information value closing delimiter in the XML declaration. |
| 30 | The parser found an invalid attribute instead of the optional encoding declaration in the XML declaration. |
| 31 | The encoding declaration value in the XML declaration was missing or incorrect. The value did not begin with lowercase or uppercase A through Z, or 'encoding' was not followed by '=', or the value was missing or improperly delimited or it specified a bad character, or the start and end delimiters did not match, or the parser found an invalid character following the closing delimiter. |
| 32 | The parser found an invalid attribute instead of the optional standalone declaration in the XML declaration. |
| 33 | The 'standalone' attribute in the XML declaration was not specified correctly. 'standalone' was not followed by a '=', or the value was either missing or improperly delimited, or the value was neither 'yes' nor 'no', or the value specified a bad character, or the start and end delimiters did not match, or the parser found an invalid character following the closing delimiter. |
| 34 | The XML declaration was not terminated by the proper character sequence '?>', or contained an invalid attribute. |
| 35 | The parser found the start of a document type declaration after the end of the root element. |
| 36 | The parser found the start of an element after the end of the root element. |
| 300 | The parser reached the end of the document before the document was complete. |
| 301 | The %HANDLER procedure for XML-INTO or XML-SAX returned a non-zero value, causing the XML parsing to end. |
| 302 | The parser does not support the requested CCSID value or the first character of the XML document was not '<'. |
| 303 | The document was too large for the parser to handle. The parser attempted to parse the incomplete document, but the data at the end of the document was necessary for the parsing to complete. |
| 500-999 | Internal error in the external parser. Please report the error to your service representative. |
| 10001-19999 | Internal error in the parser. Please report the error to your service representative. |

**0353 Parser Reason Codes**

|  |  |
| --- | --- |
| **Reason code** | **Description** |
| 1 | The specified path to the XML element was not found in the XML document. |
| 2 | The XML document contains too few array elements for array subfields of a data structure. |
|  |
| 3 | The XML document contains too many array elements for array subfields of a data structure. |
|  |
| 4 | The XML document is missing XML attributes or elements to match subfields. |
|  |
| 5 | The XML document contains extra XML attributes or elements that do not match subfields. |
|  |
| 6 | The XML document contains text content within the content for the subfields of a data structure. |
|  |
| 7 | The XML document contains unexpected attributes or child elements for XML elements that match RPG scalar fields, subfields or array elements. |
|  |
| 8 | The XML document contains data that cannot be successfully assigned to the RPG variable. The RPG status code associated with the failure is 103. |
|  |
| 9 | The XML document contains a special attribute (such as fmt or adjust) that is not relevant or does not have a valid value. |
|  |
| 10 | An XML child element was found for a data structure having a subfield matching the datasubf option. |
|  |
| 11 | An XML attribute was found having the same name as the datasubf option. |
| 12 | An XML attribute or element was found having the same name as a subfield that is identified by the countprefix option. For example, if option "countprefix=n\_" was specified, and there exist subfields "xyz" and "n\_xyz", it is not valid to have an XML attribute or element with the name "n\_xyz". |
|  |
|  |
|  |
| 13 | The countprefix and datasubf options identified the same subfield. |

**XML-SAX Event Codes**

|  |  |
| --- | --- |
| \*XML\_START\_DOCUMENT | 20 |
| \*XML\_VERSION\_INFO | 25 |
| \*XML\_ENCODING\_DECL | 10 |
| \*XML\_STANDALONE\_DECL | 18 |
| \*XML\_DOCTYPE\_DECL | 9 |
| \*XML\_START\_ELEMENT | 21 |
| \*XML\_CHARS | 5 |
| \*XML\_PREDEF\_REF | 8 |
| \*XML\_UCS2\_REF | 7 |
| \*XML\_UNKNOWN\_REF | 24 |
| \*XML\_END\_ELEMENT | 13 |
| \*XML\_ATTR\_NAME | 2 |
| \*XML\_ATTR\_CHARS | 4 |
| \*XML\_ATTR\_PREDEF\_REF | 3 |
| \*XML\_ATTR\_UCS2\_REF | 1 |
| \*XML\_UNKNOWN\_ATTR\_REF | 23 |
| \*XML\_END\_ATTR | 26 |
| \*XML\_PI\_TARGET | 16 |
| \*XML\_PI\_DATA | 17 |
| \*XML\_START\_CDATA | 19 |
| \*XML\_CHARS | 5 |
| \*XML\_END\_CDATA | 11 |
| \*XML\_COMMENT | 6 |
| \*XML\_EXCEPTION | 15 |
| \*XML\_END\_DOCUMENT | 12 |